JAVA.LANG.SECURITYMANAGER CLASS

Introduction

The **java.lang.SecurityManager** class allows applications to implement a security policy. It allows an application to determine, before performing a possibly unsafe or sensitive operation, what the operation is and whether it is being attempted in a security context that allows the operation to be performed. The application can allow or disallow the operation.

Class declaration

Following is the declaration for **java.lang.SecurityManager** class:

```
public class SecurityManager
   extends Object
```

Class constructors

S.N.	Constructor & Description
1	**SecurityManager()** This constructs a new SecurityManager.

Class methods

S.N.	Method & Description
1	void checkAccept(String host, int port) This method throws a SecurityException if the calling thread is not permitted to accept a socket connection from the specified host and port number.
2	void checkAccess(Thread t) This method throws a SecurityException if the calling thread is not allowed to modify the thread argument.
3	void checkAccess(ThreadGroup g) This method throws a SecurityException if the calling thread is not allowed to modify the thread group argument.
4	void checkAwtEventQueueAccess() This method throws a SecurityException if the calling thread is not allowed to access the AWT event queue.
5	void checkConnect(String host, int port) This method throws a SecurityException if the calling thread is not allowed to open a socket connection to the specified host and port number.
6	void checkConnect(String host, int port, Object context) This method throws a SecurityException if the specified security context is not allowed to open a socket connection to the specified host and port number.

7	void checkCreateClassLoader() This method throws a SecurityException if the calling thread is not allowed to create a new class loader.
8	void checkDelete(String file) This method throws a SecurityException if the calling thread is not allowed to delete the specified file.
9	void checkExec(String cmd) This method throws a SecurityException if the calling thread is not allowed to create a subprocess.
10	void checkExit(int status) This method throws a SecurityException if the calling thread is not allowed to cause the Java Virtual Machine to halt with the specified status code.
11	void checkLink(String lib) This method throws a SecurityException if the calling thread is not allowed to dynamic link the library code specified by the string argument file.
12	void checkListen(int port) This method throws a SecurityException if the calling thread is not allowed to wait for a connection request on the specified local port number.
13	void checkMemberAccess(Class<?> clazz, int which) This method throws a SecurityException if the calling thread is not allowed to access members.
14	void checkMulticast(InetAddress maddr) This method throws a SecurityException if the calling thread is not allowed to use (join/leave/send/receive) IP multicast.
15	void checkPackageAccess(String pkg) This method throws a SecurityException if the calling thread is not allowed to access the package specified by the argument.
16	void checkPackageDefinition(String pkg) This method throws a SecurityException if the calling thread is not allowed to define classes in the package specified by the argument.
17	void checkPermission(Permission perm) This method throws a SecurityException if the requested access, specified by the given permission, is not permitted based on the security policy currently in effect.
18	void checkPermission(Permission perm, Object context) This method throws a SecurityException if the specified security context is denied access to the resource specified by the given permission.
19	void checkPrintJobAccess() This method throws a SecurityException if the calling thread is not allowed to initiate a print job request.
20	void checkPropertiesAccess() This method throws a SecurityException if the calling thread is not allowed to access or modify the system properties.
21	void checkPropertyAccess(String key) This method throws a SecurityException if the calling thread is not allowed to access the system property with the specified key name.
22	void checkRead(FileDescriptor fd) This method throws a SecurityException if the calling thread is not allowed to read from the specified file

	descriptor.
23	void checkRead(String file) This method throws a SecurityException if the calling thread is not allowed to read the file specified by the string argument.
24	void checkRead(String file, Object context) This method throws a SecurityException if the specified security context is not allowed to read the file specified by the string argument.
25	void checkSecurityAccess(String target) This method determines whether the permission with the specified permission target name should be granted or denied.
26	void checkSetFactory() This method throws a SecurityException if the calling thread is not allowed to set the socket factory used by ServerSocket or Socket, or the stream handler factory used by URL.
27	void checkSystemClipboardAccess() This method throws a SecurityException if the calling thread is not allowed to access the system clipboard.
28	boolean checkTopLevelWindow(Object window) This method returns false if the calling thread is not trusted to bring up the top-level window indicated by the window argument.
29	void checkWrite(FileDescriptor fd) This method throws a SecurityException if the calling thread is not allowed to write to the specified file descriptor.
30	void checkWrite(String file) This method throws a SecurityException if the calling thread is not allowed to write to the file specified by the string argument.
31	protected Class[] getClassContext() This method returns the current execution stack as an array of classes.
32	Object getSecurityContext() This method creates an object that encapsulates the current execution environment.
33	ThreadGroup getThreadGroup() This method returns the thread group into which to instantiate any new thread being created at the time this is being called.

Methods inherited

This class inherits methods from the following classes:

- java.lang.Object